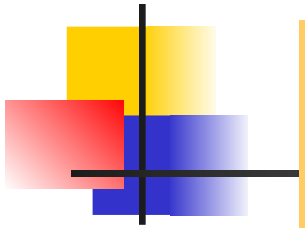
Plan du cours

- ✓ Introduction
- ✓ Modèle entité-association
- ✓ Modèle relationnel
- ■ Langage SQL

Langage SQL

- **SQL** = Structured Query Language
- **SQL** permet la définition, la manipulation et le contrôle d'une base de données relationnelle. Il se base sur l'algèbre relationnelle.
- **SQL** est un standard depuis 1986
- On adoptera dans ce cours la syntaxe d' **Oracle**

Les Ordres SQL



SELECT	**Recherche de données**
INSERT UPDATE DELETE	**Langage de manipulation des données (LMD)**
CREATE ALTER DROP RENAME TRUNCATE	**Langage de définition des données (LDD)**
COMMIT ROLLBACK	**Contrôle des transactions**
GRANT REVOKE	**Langage de contrôle des données (LCD)**

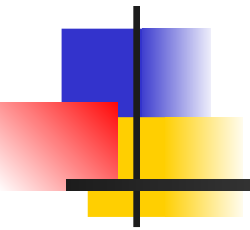
Tables utilisées dans ce cours

EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	1500		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON



Création et Gestion de Tables

Conventions de Dénomination

Un nom :

- Doit commencer par une lettre
- Peut comporter de 1 à 30 caractères
- Ne peut contenir que les caractères A à Z, a à z, 0 à 9, _, $, et #
- Ne doit pas porter le nom d'un autre objet appartenant au même utilisateur
- Ne doit pas être un mot réservé Oracle Server

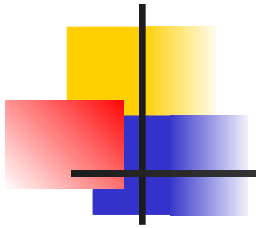
L'Ordre CREATE TABLE

- Vous devez posséder :
  - Un privilège CREATE TABLE
  - Un espace de stockage

```
CREATE TABLE [schema.]table
             (column datatype [DEFAULT expr],...
```

- Spécifiez :
  - Un nom de table
  - Le nom, le type de données et la taille des colonnes.

Références aux Tables d'un Autre Utilisateur



- Les tables appartenant à d'autres utilisateurs ne sont pas dans le schéma utilisateur.
- Le nom du propriétaire doit précéder le nom de la table.

L'Option DEFAULT

- Spécifie la valeur par défaut d'une colonne

```
… hiredate DATE DEFAULT SYSDATE, …
```

- Valeurs autorisées : littéraux, expressions et fonctions SQL.
- Valeurs non-autorisées : noms d'autres colonnes ou pseudo - colonnes.
- Le type de données par défaut doit correspondre à celui de la colonne.

Création de Tables

- Créer la table.

```
SQL> CREATE TABLE service
  2          (numserv        CHAR(6),
  3           nomserv        VARCHAR2(15));
Table créée.
```

- Vérifier la création de la table.

```
SQL> DESCRIBE service
```

```
SQL> desc service
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------
 NUMSERV                                   NOT NULL CHAR(6)
 NOMSERV                                            VARCHAR2(14)
```

Types de Données

Types de données	Description
VARCHAR2(*size*)	Données caractères de longueur variable
CHAR(*size*)	Données caractères de longueur fixe
NUMBER(*p*,*s*)	Numérique de longueur variable
DATE	Valeurs de date et d'heure
LONG	Données caractères de longueur variable, jusqu'à 2 giga-octets

L'ordre ALTER TABLE

- Utilisez l'ordre ALTER TABLE pour :
  - Ajouter une colonne
  - Modifier une colonne existante
  - Définir une valeur par défaut pour une nouvelle colonne

```
ALTER TABLE table
ADD         (column datatype [DEFAULT expr]
            [, column datatype]...);
```

```
ALTER TABLE table
MODIFY      (column datatype [DEFAULT expr]
            [, column datatype]...);
```

Ajout de Colonnes

SERVICE20

MATRICULE	NOM	SAL_ANN	DATE_EMB
------	----------	--------	
e5	CLERCK	48000	12/01/95
e9	FORD	20400	01/01/94
...			

Nouvelle colonne

FONCTION

"...ajouter une nouvelle colonne à la table SERVICE20..."

SERVICE20

MATRICULE	NOM	SAL_ANN	DATE_EMB	FONCTION
------	----------	--------		
e5	CLERCK	48000	12/01/95	
e9	FORD	20400	01/01/94	
...				

Ajout de Colonnes

- Utilisez la clause ADD pour ajouter des colonnes.

```
SQL> ALTER TABLE service20
  2  ADD          (fonction VARCHAR2(9));
Table modifiée.
```

- La nouvelle colonne est placée à la fin.

```
MATRIC NOM                  SAL_ANN DATE_EMB FONCTION
------ --------------- ---------- -------- ---------
e5     Clerck                48000 12/01/95
e9     Ford                  20400 01/01/94
```

Modification de Colonnes

- Vous pouvez modifier le type de données, la taille et la valeur par défaut d'une colonne.

```
ALTER TABLE  service20
MODIFY (nom VARCHAR2(20));
Table modifiée.
```

- la modification d’une valeur par défaut ne s’applique qu’aux insertions ultérieures dans la table.

Suppression de Tables

- La structure et toutes les données de la table sont supprimées.
- Tous les index sont supprimés.
- La transaction en instance est validée.
- Une suppression de table ne peut être annulée.

```
SQL> DROP TABLE service20;
Table supprimée.
```

Modification du Nom d'un Objet

- Pour modifier le nom d'une table, d'une vue, d'une séquence ou d'un synonyme, utilisez l'ordre RENAME.

```
SQL> RENAME service TO departement;
Table renommée.
```

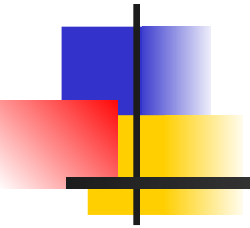
- Vous devez être propriétaire de l'objet.

Vider une Table

- L'ordre TRUNCATE TABLE :
  - Supprime toutes les lignes d'une table
  - Libère l'espace de stockage utilisé par la

```
SQL> TRUNCATE TABLE departement;
Table tronquée.
```

- Vous ne pouvez pas annuler un ordre TRUNCATE
- Vous pouvez aussi utiliser l'ordre DELETE pour supprimer des lignes



Les Contraintes

Les Contraintes

- Les contraintes contrôlent des règles de gestion au niveau d'une table.
- Les contraintes empêchent la suppression d'une table lorsqu'il existe des dépendances.
- Types de contraintes valides dans Oracle :
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY

Les Contraintes

```
CREATE TABLE [schema.]table
             (column datatype [DEFAULT expr]
              [column_constraint],
              …
              [table_constraint]);
```

```
CREATE TABLE salarie(
              matricule  CHAR(6),
              nom        VARCHAR2(15),
              …
              numserv     CHAR(6) NOT NULL,
              CONSTRAINT salarie_matricule_pk
                           PRIMARY KEY (MATRICULE));
```

Les Contraintes

- Contrainte au niveau colonne

```
column [CONSTRAINT constraint_name] constraint_type,
```

- Contrainte au niveau table

```
column,...
  [CONSTRAINT constraint_name] constraint_type
  (column, ...),
```

La Contrainte NOT NULL

- Interdit la présence de valeurs NULL dans la colonne

SALARIE

MATRIC	NOM	FONCTION	...	COMM	NUMSERV
e1	KING	PRESIDENT			s10
e2	SMITH	CADRE			s30
e3	FORD	ANALYSTE			s10
e4	CLERCK	CADRE			s20
...					

Contrainte NOT NULL (aucune ligne ne peut avoir de valeur NULL dans cette colonne)

Absence de contrainte NOT NULL (toute ligne peut avoir une valeur NULL dans cette colonne)

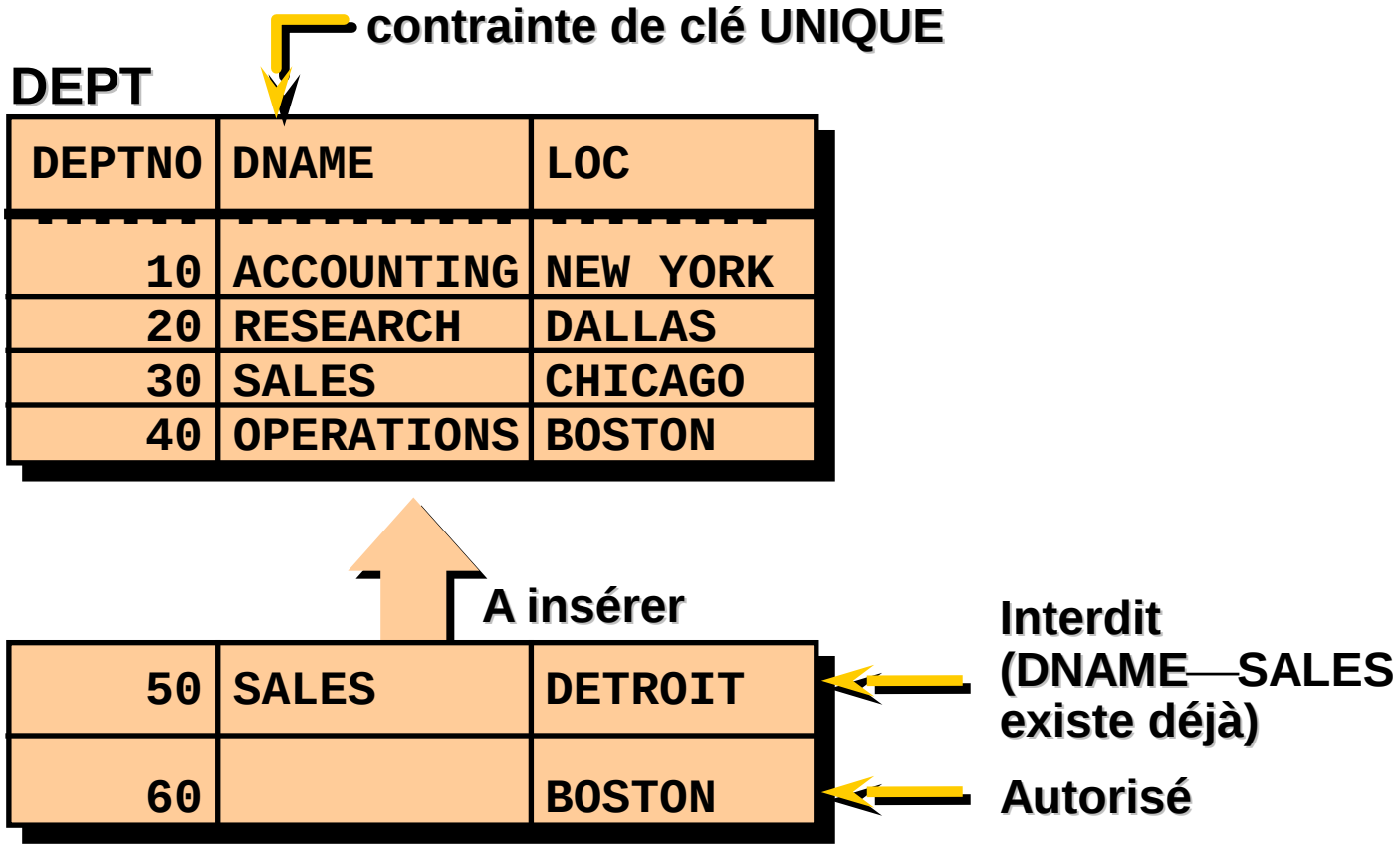
Contrainte NOT NULL

La Contrainte NOT NULL

- Se définit au niveau colonne

```
SQL> CREATE TABLE salarie(
  2      matricule   CHAR(6),
  3      nom         VARCHAR2(15) NOT NULL,
  4      fonction    VARCHAR2(9),
  5      date_emb    DATE,
  6      salaire     NUMBER(7,2),
  7      comm        NUMBER(7,2),
  8      numserv     CHAR(6) NOT NULL);
```

La Contrainte de Clé UNIQUE

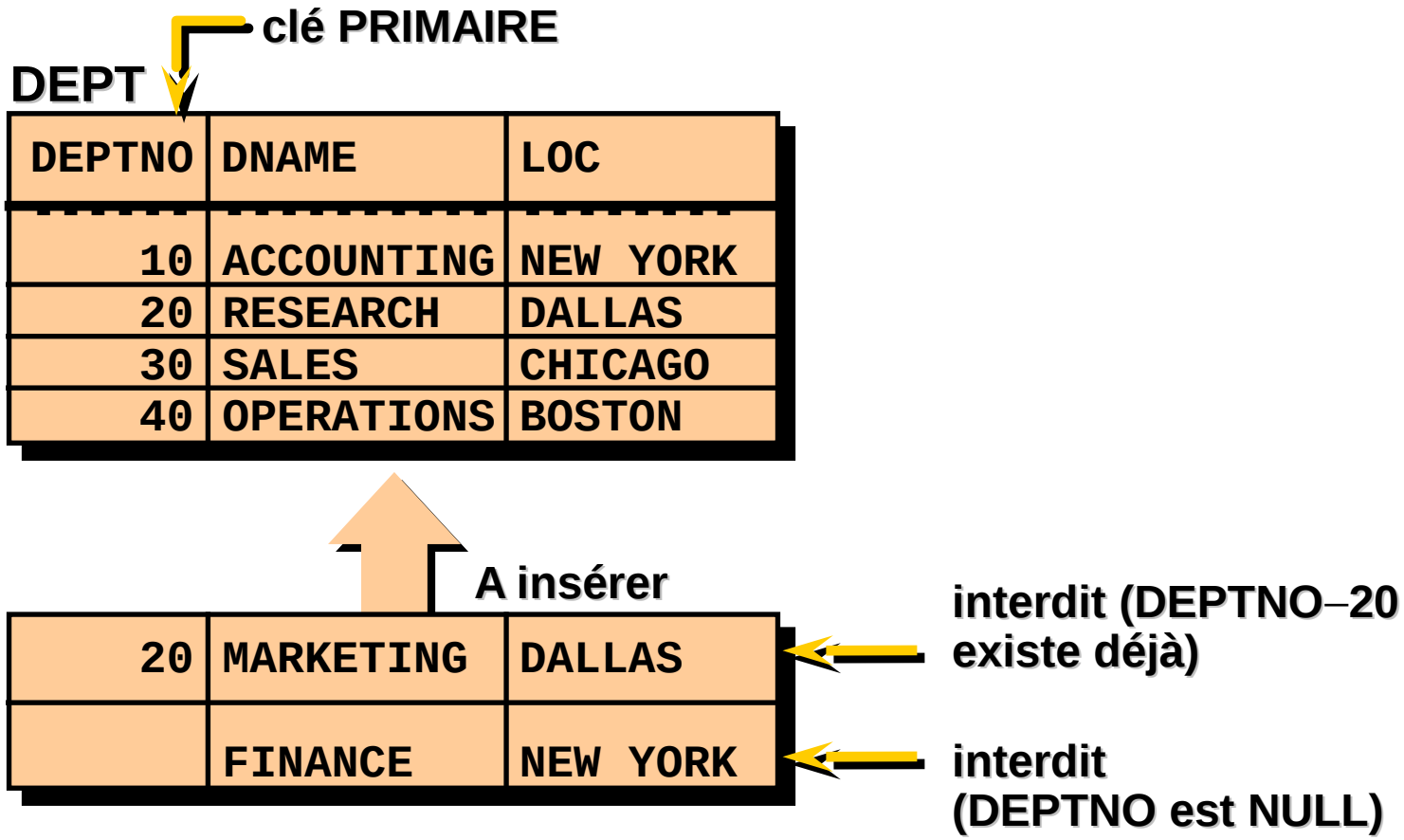


La Contrainte de Clé UNIQUE

- Se définit au niveau table ou colonne

```
SQL> CREATE TABLE dept(
  2       deptno    NUMBER(2),
  3       dname     VARCHAR2(14),
  4       loc       VARCHAR2(13),
  5       CONSTRAINT dept_dname_uk UNIQUE(dname));
```

La Contrainte PRIMARY KEY

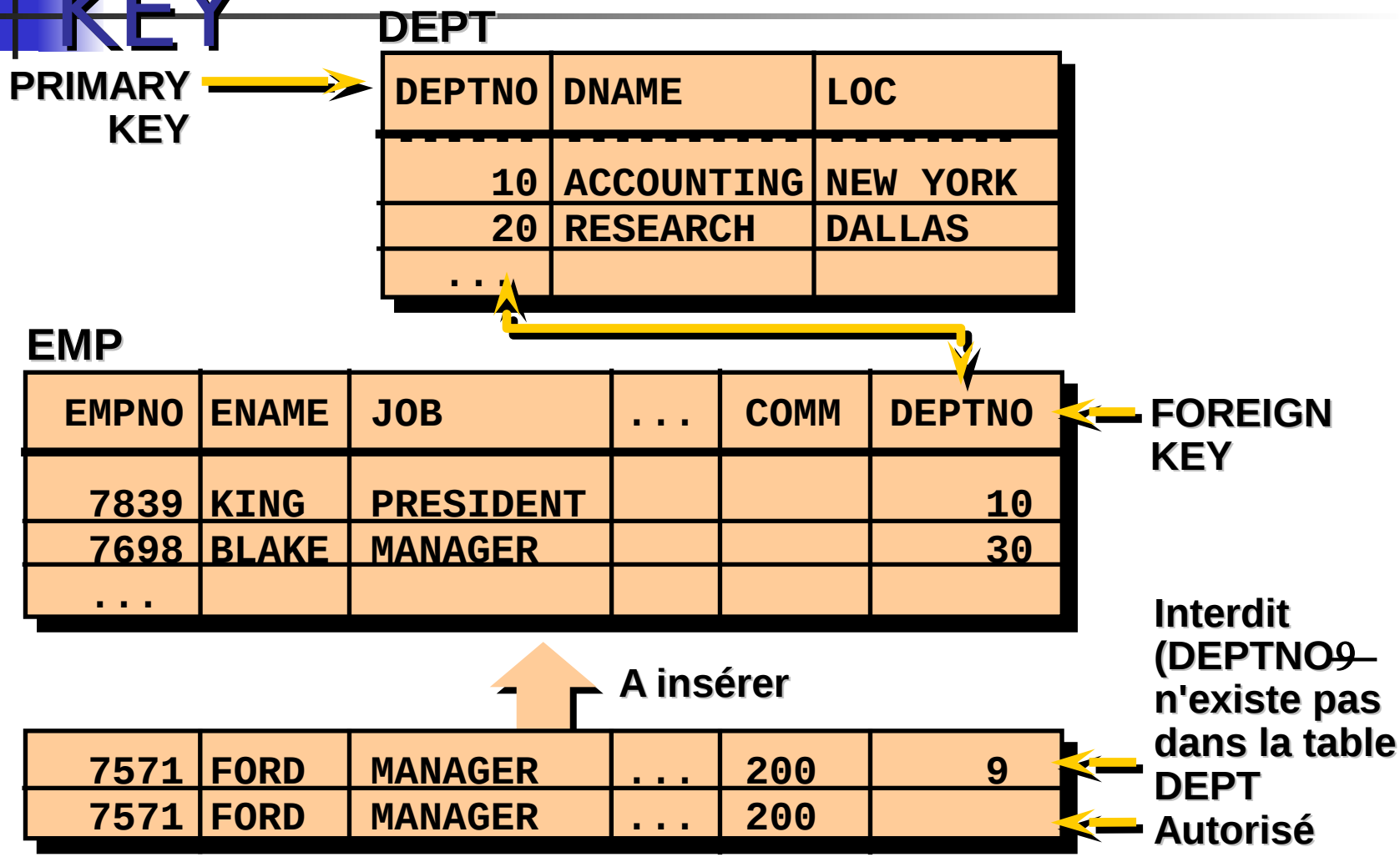


La Contrainte PRIMARY KEY

- Se définit au niveau table ou colonne

```
SQL> CREATE TABLE   dept(
  2       deptno      NUMBER(2),
  3       dname       VARCHAR2(14),
  4       loc         VARCHAR2(13),
  5       CONSTRAINT dept_dname_uk UNIQUE (dname),
  6       CONSTRAINT dept_deptno_pk PRIMARY KEY(deptno));
```

La Contrainte FOREIGN KEY



La Contrainte FOREIGN KEY

- Se définit au niveau table ou

```
SQL> CREATE TABLE emp(
  2       empno    NUMBER(4),
  3       ename    VARCHAR2(10) NOT NULL,
  4       job      VARCHAR2(9),
  5       mgr      NUMBER(4),
  6       hiredate DATE,
  7       sal      NUMBER(7,2),
  8       comm     NUMBER(7,2),
  9       deptno   NUMBER(2) NOT NULL,
 10       CONSTRAINT emp_deptno_fk FOREIGN KEY (deptno)
 11                  REFERENCES dept (deptno));
```

Mots-clés Associés à la Contrainte FOREIGN KEY

- FOREIGN KEY
  - Définit la colonne dans la table détail dans une contrainte de niveau table
- REFERENCES
  - Identifie la table et la colonne de la table maître
- ON DELETE CASCADE
  - Autorise la suppression d’une ligne dans la table maître et des lignes dépendantes dans la table détail

La Contrainte CHECK

- Définit une condition que chaque ligne doit obligatoirement satisfaire

```
..., deptno   NUMBER(2),
      CONSTRAINT emp_deptno_ck
            CHECK (DEPTNO BETWEEN 10 AND 99),...
```

Ajout d'une Contrainte

```
ALTER TABLE table
ADD [CONSTRAINT constraint] type (column);
```

- Vous pouvez ajouter ou supprimer une contrainte, mais pas la modifier
- Vous pouvez activer ou désactiver des contraintes
- Pour ajouter une contrainte NOT NULL, utilisez la clause MODIFY

Ajout d'une Contrainte

- Ajouter une contrainte FOREIGN KEY à la table EMP précisant qu'un manager doit déjà exister dans la table EMP en tant qu'employé valide.

```
SQL> ALTER TABLE    emp
  2  ADD CONSTRAINT emp_mgr_fk
  3            FOREIGN KEY(mgr) REFERENCES emp(empno);
Table altered.
```

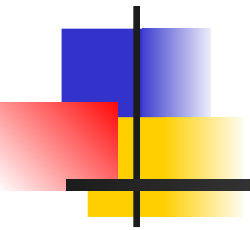
Suppression d'une Contrainte

- Supprimer de la table EMP la contrainte concernant le manager.

```
SQL> ALTER TABLE          emp
  2  DROP CONSTRAINT      emp_mgr_fk;
Table altered.
```

- Supprimer la contrainte PRIMARY KEY de la table DEPT, ainsi que la contrainte FOREIGN KEY associée définie sur la colonne EMP.DEPTNO.

```
SQL> ALTER TABLE          dept
  2  DROP PRIMARY KEY CASCADE;
Table altered.
```



L'Ordre **SELECT**

Élémentaire

Les Possibilités de l'Ordre SQL **SELECT**

Sélection

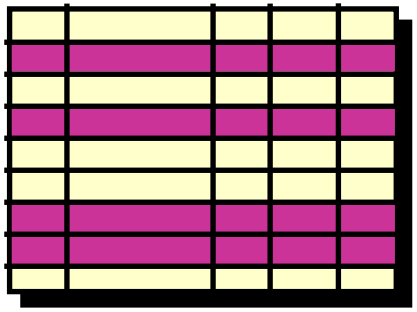


Table 1

Projection

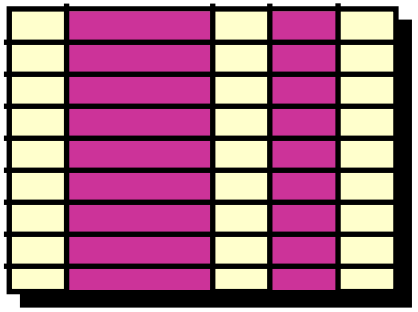


Table 1

Jointure

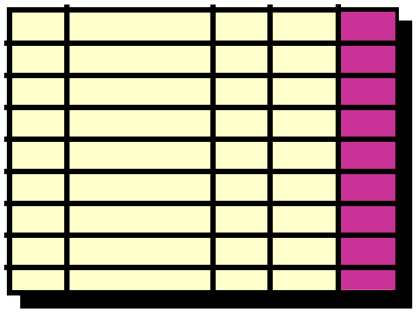


Table 1

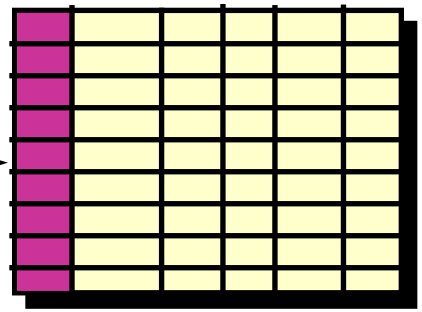
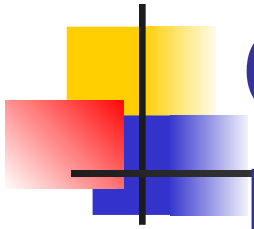


Table 2

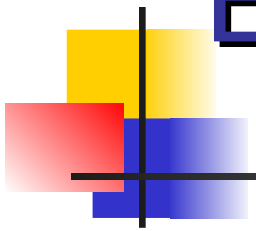


Ordre **SELECT** Élémentaire

```
SELECT   [DISTINCT] {*, column [alias],...}
FROM     table;
```

- SELECT indique *quelles* colonnes rapporter
- FROM indique dans *quelle* table rechercher

Écriture des Ordres SQL



- Les ordres SQL peuvent être écrits indifféremment en majuscules et/ou minuscules.
- Les ordres SQL peuvent être écrits sur plusieurs lignes.
- Les mots-clés ne doivent pas être abrégés ni scindés sur deux lignes différentes.
- Les clauses sont généralement placées sur des lignes distinctes.
- Les tabulations et indentations

Sélection de Toutes les Colonnes

```
SQL> SELECT *
  2  FROM   dept;
```

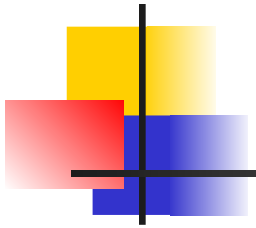
```
    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

Sélection d'Une ou Plusieurs Colonnes Spécifiques

```
SQL> SELECT deptno, loc
  2  FROM   dept;
```

```
    DEPTNO LOC
---------- -------------
        10 NEW YORK
        20 DALLAS
        30 CHICAGO
        40 BOSTON
```

Valeurs par Défaut des En-têtes de Colonne



- Justification par défaut
  - A gauche : date et données alphanumériques
  - A droite : données numériques
- Affichage par défaut : en majuscules

Expressions Arithmétiques

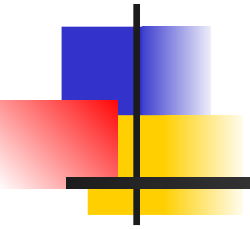
- Possibilité de créer des expressions avec des données de type NUMBER et DATE au moyen d'opérateurs arithmétiques

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division

Utilisation des Opérateurs Arithmétiques

```
SQL> SELECT ename, sal, sal+300
  2  FROM   emp;
```

```
ENAME                SAL     SAL+300
---------- --------- ---------
KING                5000        5300
BLAKE               2850        3150
CLARK               2450        2750
JONES               2975        3275
MARTIN              1250        1550
ALLEN               1600        1900
...
14 rows selected.
```



Sélection et Tri des Lignes Retournées par un SELECT

Sélectionner les Lignes

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				

"...rechercher tous les employés du département 10"

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7782	CLARK	MANAGER		10
7934	MILLER	CLERK		10

Sélectionner les Lignes

- Restreindre la sélection au moyen de la clause WHERE.

```
SELECT      [DISTINCT] {*, column [alias], ...}
FROM        table
[WHERE      condition(s)];
```

- La clause WHERE se place après la clause FROM.

Utilisation de la Clause WHERE

```
SQL> SELECT ename, job, deptno
  2  FROM   emp
  3  WHERE  job='CLERK';
```

```
ENAME      JOB          DEPTNO
---------- --------- ---------
JAMES      CLERK            30
SMITH      CLERK            20
ADAMS      CLERK            20
MILLER     CLERK            10
```

Chaînes de Caractères et Dates

- Les constantes chaînes de caractères et dates doivent être placées entre simples quotes.
- La recherche tient compte des majuscules et minuscules (pour les chaînes de caractère) et du format (pour les dates.)
- Le format de date par défaut est 'DD-MON-YY'.

```
SQL> SELECT   ename, job, deptno
  2  FROM     emp
  3  WHERE    ename = 'JAMES';
```

Opérateurs de Comparaison

Opérateur	Signification
=	Egal à
>	Supérieur à
>=	Supérieur ou égal à
<	Inférieur à
<=	Inférieur ou égal à
<>	Différent de

Utilisation des Opérateurs de Comparaison

```
SQL> SELECT ename, sal, comm
  2  FROM   emp
  3  WHERE  sal<=comm;
```

```
ENAME           SAL       COMM
---------- ---------- ----------
MARTIN          1250 <--> 1400
```

Autres Opérateurs de Comparaison

Opérateur	Signification
BETWEEN ...AND...	Compris entre ... et ... (bornes comprises)
IN (liste)	Correspond à une valeur de la liste
LIKE	Ressemblance partielle de chaînes de caractères
IS NULL	Correspond à une valeur NULL

Utilisation de l'Opérateur BETWEEN

- BETWEEN permet de tester l'appartenance à une fourchette de valeurs.

```
SQL> SELECT  ename, sal
  2  FROM    emp
  3  WHERE   sal BETWEEN 1000 AND 1500;
```

ENAME	SAL
MARTIN	1250
TURNER	1500
WARD	1250
ADAMS	1100
MILLER	1300

Limite inférieure

Limite supérieure

Utilisation de l'Opérateur IN

- IN permet de comparer une expression avec une liste de valeurs.

```
SQL> SELECT   empno, ename, sal, mgr
  2  FROM     emp
  3  WHERE    mgr IN (7902, 7566, 7788);
```

```
     EMPNO ENAME                SAL        MGR
---------- ---------- ---------- ----------
      7902 FORD                3000       7566
      7369 SMITH                800       7902
      7788 SCOTT               3000       7566
      7876 ADAMS               1100       7788
```

Utilisation de l'Opérateur LIKE

- LIKE permet de rechercher des chaînes de caractères à l'aide de caractères génériques
- Les conditions de recherche peuvent contenir des caractères ou des nombres littéraux.
  - (%) représente zéro ou plusieurs caractères
  - (_) représente un caractère

```
SQL> SELECT   ename
  2  FROM     emp
  3  WHERE    ename LIKE 'S%';
```

Utilisation de l'Opérateur LIKE

- Vous pouvez combiner plusieurs caractères génériques de recherche.

```
SQL> SELECT   ename
  2  FROM     emp
  3  WHERE    ename LIKE '_A%';
```

```
ENAME
----------
JAMES
WARD
```

- Vous pouvez utiliser l'identifiant ESCAPE pour rechercher "%" ou "_".

Utilisation de l'Opérateur IS NULL

- Recherche de valeurs NULL avec l'opérateur IS NULL

```
SQL> SELECT  ename, mgr
  2  FROM    emp
  3  WHERE   mgr IS NULL;
```

```
ENAME      MGR
---------- ---------
KING
```

Clause ORDER BY

- Tri des lignes avec la clause ORDER BY
  - ASC : ordre croissant (par défaut)
  - DESC : ordre décroissant
- La clause ORDER BY se place à la fin de l'ordre SELECT

```
SQL> SELECT    ename, job, deptno, hiredate
  2  FROM      emp
  3  ORDER BY hiredate;
```

```
ENAME      JOB              DEPTNO HIREDATE
---------- --------- ---------- ---------
SMITH      CLERK            20 17-DEC-80
ALLEN      SALESMAN         30 20-FEB-81
...
14 rows selected.
```

Tri par Ordre Décroissant

```
SQL> SELECT    ename, job, deptno, hiredate
  2  FROM      emp
  3  ORDER BY hiredate DESC;
```

```
ENAME      JOB              DEPTNO HIREDATE
---------- --------- --------- ---------
ADAMS      CLERK              20 12-JAN-83
SCOTT      ANALYST            20 09-DEC-82
MILLER     CLERK              10 23-JAN-82
JAMES      CLERK              30 03-DEC-81
FORD       ANALYST            20 03-DEC-81
KING       PRESIDENT          10 17-NOV-81
MARTIN     SALESMAN           30 28-SEP-81
...
14 rows selected.
```

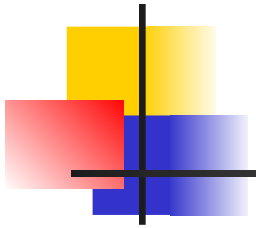
Tri sur Plusieurs Colonnes

- L'ordre des éléments de la liste ORDER BY donne l'ordre du tri.

```
SQL> SELECT   ename, deptno, sal
  2  FROM     emp
  3  ORDER BY  deptno, sal DESC;
```

```
ENAME          DEPTNO        SAL
---------- --------- ---------
KING               10       5000
CLARK              10       2450
MILLER             10       1300
FORD               20       3000
...
14 rows selected.
```

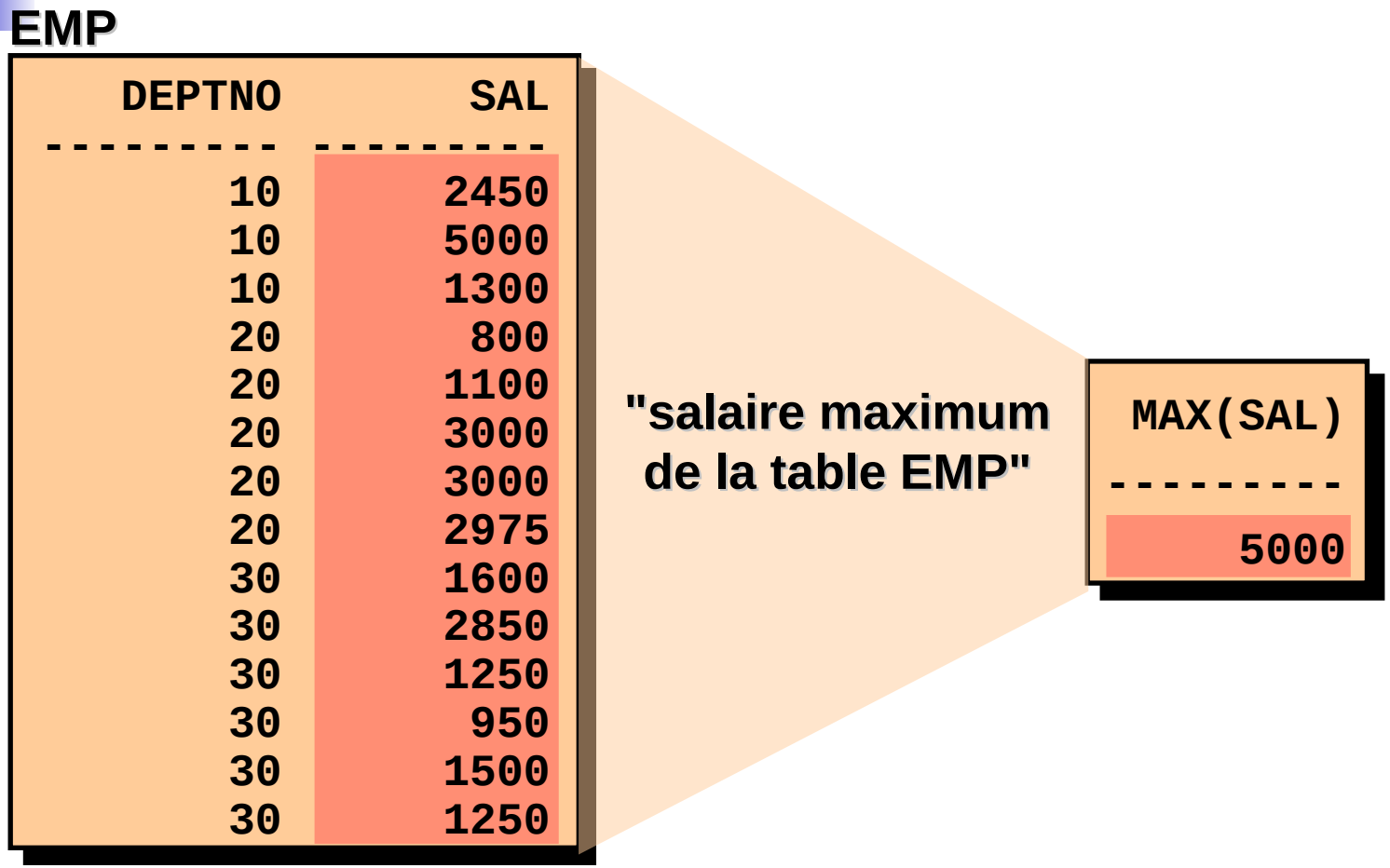
- Vous pouvez effectuer un tri sur une colonne ne figurant pas dans la liste SELECT.



Regrouper les Données avec les Fonctions de Groupe

Fonctions de Groupe

- Les fonctions de groupe agissent sur des groupes de lignes et donnent un résultat par groupe.



Types de Fonctions de Groupe

- AVG ([DISTINCT|ALL]*n*)
- COUNT ({ *|[DISTINCT|ALL]*expr*})
- MAX ([DISTINCT|ALL]*expr*)
- MIN ([DISTINCT|ALL]*expr*)
- SUM ([DISTINCT|ALL]*n*)
- VARIANCE ([DISTINCT|ALL]*n*)

Fonctions AVG et SUM

- AVG et SUM s'utilisent avec des données numériques.

```
SQL> SELECT   AVG(sal), MAX(sal),
  2           MIN(sal), SUM(sal)
  3  FROM     emp
  4  WHERE    job LIKE 'SALES%';
```

AVG(SAL)	MAX(SAL)	MIN(SAL)	SUM(SAL)
1400	1600	1250	5600

Fonctions MIN et MAX

- MIN et MAX s'utilisent avec tous types de données.

```
SQL> SELECT   MIN(hiredate), MAX(hiredate)
  2  FROM     emp;
```

```
MIN(HIRED MAX(HIRED
--------- ---------
17-DEC-80 12-JAN-83
```

Utilisation de la Fonction COUNT

- COUNT(*) ramène le nombre de lignes d'une table.

```
SQL> SELECT   COUNT(*)
  2  FROM     emp
  3  WHERE    deptno = 30;
```

```
  COUNT(*)
----------
         6
```

Utilisation de la Fonction COUNT

- COUNT(*expr*) ramène le nombre de lignes non NULL.

```
SQL> SELECT   COUNT(comm)
  2  FROM     emp
  3  WHERE    deptno = 30;
```

```
COUNT(COMM)
-----------
          4
```

Création de Groupes de Données

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

2916.6667

2175

1566.6667

"salaire moyen pour chaque département de la table EMP"

DEPTNO	AVG(SAL)
10	2916.6667
20	2175
30	1566.6667

Création de Groupes de Données : la Clause GROUP BY

```
SELECT        column, group_function
FROM          table
[WHERE        condition]
[GROUP BY     group_by_expression]
[ORDER BY     column];
```

- Divisez une table en groupes de lignes avec la clause GROUP BY.

Utilisation de la Clause GROUP BY

- La clause GROUP BY doit inclure toutes les colonnes de la liste SELECT qui ne figurent pas dans des fonctions de groupe.

```
SQL> SELECT   deptno, AVG(sal)
  2  FROM     emp
  3  GROUP BY deptno;
```

```
   DEPTNO  AVG(SAL)
--------- ---------
       10 2916.6667
       20       2175
       30 1566.6667
```

Utilisation de la Clause GROUP BY

La colonne citée en GROUP BY ne doit pas nécessairement figurer dans la liste SELECT.

```
SQL> SELECT   AVG(sal)
  2  FROM     emp
  3  GROUP BY deptno;
```

```
 AVG(SAL)
---------
2916.6667
      2175
1566.6667
```

Regroupement sur Plusieurs Colonnes

EMP

DEPTNO	JOB	SAL
10	MANAGER	2450
10	PRESIDENT	5000
10	CLERK	1300
20	CLERK	800
20	CLERK	1100
20	ANALYST	3000
20	ANALYST	3000
20	MANAGER	2975
30	SALESMAN	1600
30	MANAGER	2850
30	SALESMAN	1250
30	CLERK	950
30	SALESMAN	1500
30	SALESMAN	1250

'"somme des salaires de la table EMP pour chaque poste, regroupés par département"

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	6000
20	CLERK	1900
20	MANAGER	2975
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	5600

Utilisation de la Clause GROUP BY sur Plusieurs Colonnes

```
SQL> SELECT   deptno, job, sum(sal)
  2  FROM     emp
  3  GROUP BY deptno, job;
```

```
    DEPTNO JOB         SUM(SAL)
---------- --------- ---------
        10 CLERK          1300
        10 MANAGER        2450
        10 PRESIDENT      5000
        20 ANALYST        6000
        20 CLERK          1900
...
9 rows selected.
```

Erreurs d'utilisation des Fonctions de Groupe dans une Requête

- Vous ne pouvez utiliser la clause WHERE pour limiter les groupes.
- Utilisez la clause HAVING.

```
SQL> SELECT    deptno, AVG(sal)
  2  FROM      emp
  3  WHERE     AVG(sal) > 2000
  4  GROUP BY  deptno;
```

```
WHERE AVG(sal) > 2000
      *
ERROR at line 3:
ORA-00934: group function is not allowed here
```

N'utilisez pas la clause WHERE pour limiter les groupes

Exclusion de Groupes

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

5000

3000

2850

"salaire maximum supérieur à $2900 dans chaque département"

DEPTNO	MAX(SAL)
10	5000
20	3000

Exclusion de Groupes : la Clause HAVING

- Utilisez la clause HAVING pour restreindre les groupes
  - Les lignes sont regroupées.
  - La fonction de groupe est appliquée.
  - Les groupes qui correspondent à la clause HAVING sont affichés.

```
SELECT        column, group_function
FROM          table
[WHERE        condition]
[GROUP BY     group_by_expression]
[HAVING       group_condition]
[ORDER BY     column];
```

Utilisation de la clause HAVING

```
SQL> SELECT   deptno, max(sal)
  2  FROM     emp
  3  GROUP BY deptno
  4  HAVING   max(sal)>2900;
```

```
    DEPTNO  MAX(SAL)
---------- ---------
        10      5000
        20      3000
```

Utilisation de la clause HAVING

```
SQL> SELECT    job, SUM(sal) PAYROLL
  2  FROM      emp
  3  WHERE      job NOT LIKE 'SALES%'
  3  GROUP BY  job
  4  HAVING    SUM(sal)>5000
  5  ORDER BY  SUM(sal);
```

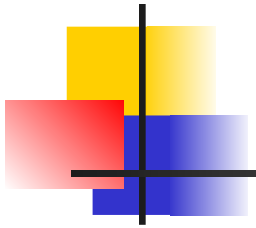
```
JOB          PAYROLL
--------- ---------
ANALYST         6000
MANAGER         8275
```

Imbrication des Fonctions de Groupe

- Afficher le salaire moyen maximum.

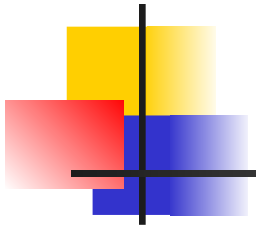
```
SQL> SELECT   max(avg(sal))
  2  FROM     emp
  3  GROUP BY deptno;
```

```
MAX(AVG(SAL))
-------------
    2916.6667
```



Manipulation des Données

Langage de Manipulation des Données



- Un ordre du LMD est exécuté lorsque :
  - Vous ajoutez des lignes à une table
  - Vous modifiez des lignes existantes dans une table
  - Vous supprimez des lignes d'une table
- Une *transaction* est un ensemble d'ordres du LMD formant une unité de travail logique.

Ajout d'une Nouvelle Ligne dans une Table

50	DEVELOPMENT	DETROIT

Nouvelle ligne

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

"...insérer une nouvelle ligne dans la table DEPT ..."

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT

L'Ordre INSERT

- L'ordre INSERT permet d'ajouter de nouvelles lignes dans une table.

```
INSERT INTO          table [(column [, column...])]
VALUES               (value [, value...]);
```

- Cette syntaxe n'insère qu'une seule ligne à la fois.

Insertion de Nouvelles Lignes

- Insérez une nouvelle ligne en précisant une valeur pour chaque colonne.
- Eventuellement, énumérez les colonnes dans la clause INSERT.

```
SQL> INSERT INTO dept(deptno, dname, loc)
  2  VALUES          (50, 'DEVELOPMENT', 'DETROIT');
1 row created.
```

- Indiquez les valeurs dans l'ordre par défaut des colonnes dans la table.
- Placez les valeurs de type caractère et date entre simples quotes.

Copie de Lignes d'une Autre Table

- Ecrivez votre ordre INSERT en spécifiant une sous-interrogation.

```
SQL> INSERT INTO managers(id, name, salary, hiredate)
  2                 SELECT empno, ename, sal, hiredate
  3                 FROM   emp
  4                 WHERE  job = 'MANAGER';
3 rows created.
```

- N'utilisez pas la clause VALUES.
- Le nombre de colonnes de la clause INSERT doit correspondre à celui de la sous-interrogation.

Modification des Données d'une Table

EMP

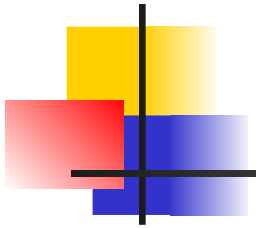
EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				

"...modifier une ligne de la table EMP..."

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		20
7566	JONES	MANAGER		20
...				

L'Ordre UPDATE



- Utilisez l'ordre UPDATE pour modifier des lignes existantes.

```
UPDATE      table
SET         column = value [, column = value]
[WHERE      condition];
```

- Si nécessaire, vous pouvez modifier plusieurs lignes à la fois.

Modification de Lignes d'une Table

- La clause WHERE permet de modifier une ou plusieurs lignes spécifiques.

```
SQL> UPDATE   emp
  2  SET      deptno = 20
  3  WHERE    empno = 7782;
1 row updated.
```

- Si vous omettez la clause WHERE, toutes les lignes sont modifiées

```
SQL> UPDATE   employee
  2  SET      deptno = 20;
14 rows updated.
```

Modification de Lignes : Erreur de Contrainte d'Intégrité

```
SQL> UPDATE   emp
  2  SET      deptno = 55
  3  WHERE    deptno = 10;
```

```
UPDATE emp
       *
ERROR at line 1:
ORA-02291: integrity constraint (USR.EMP_DEPTNO_FK)
violated - parent key not found
```

- Le numéro de département 55 n'existe pas

Suppression d'une Ligne d'une Table

DEPT

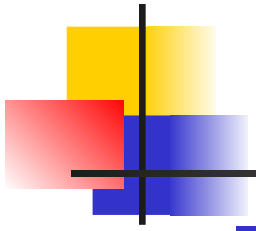
DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT
60	MIS	
...		

"...supprime une ligne de la table DEPT..."

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
60	MIS	
...		

L'Ordre DELETE



- Vous pouvez supprimer des lignes d'une table au moyen de l'ordre DELETE.

```
DELETE [FROM]   table
[WHERE          condition];
```

Suppression de Lignes d'une Table

- La clause WHERE permet de supprimer une ou plusieurs lignes spécifiques.

```
SQL> DELETE FROM    department
  2  WHERE           dname = 'DEVELOPMENT';
1 row deleted.
```

Si vous omettez la clause WHERE, toutes les lignes sont supprimées.

```
SQL> DELETE FROM    department;
4 rows deleted.
```

Suppression de Lignes : Erreur de Contrainte d'Intégrité

```
SQL> DELETE FROM    dept
  2  WHERE          deptno = 10;
```

```
DELETE FROM dept
            *
ERROR at line 1:
ORA-02292: integrity constraint (USR.EMP_DEPTNO_FK)
violated - child record found
```

- Vous ne pouvez pas supprimer une ligne qui contient une clé primaire utilisée comme clé étrangère dans une autre table.

Transactions de Base de Données

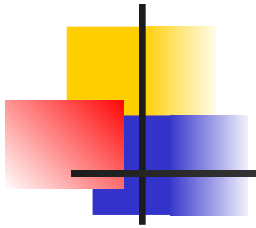
- Une transaction se compose des éléments suivants :
  - Ensemble d'ordres du LMD effectuant une modification cohérente des données
  - Un ordre du LDD
  - Un ordre du LCD

Transactions de Base de Données

Une transaction :

- Commence à l'exécution du premier ordre SQL
- Se termine par l'un des événements suivants :
  - COMMIT ou ROLLBACK
  - Exécution d'un ordre LDD ou LCD (validation automatique)
  - Fin de session utilisateur
  - Panne du système

Etat des Données Avant COMMIT ou ROLLBACK



- Il est possible de restaurer l'état précédent des données.
- L'utilisateur courant peut afficher le résultat des opérations du LMD au moyen de l'ordre SELECT.
- Les résultats des ordres du LMD exécutés par l'utilisateur courant *ne peuvent pas* être affichés par d'autres utilisateurs.
- Les lignes concernées sont *verrouillées*.

Etat des Données Après COMMIT

- Les modifications des données dans la base sont définitives.
- L'état précédent des données est irrémédiablement perdu.
- Tous les utilisateurs peuvent voir le résultat des modifications.
- Les lignes verrouillées sont libérées et peuvent de nouveau être manipulées par d'autres utilisateurs.

Validation de Données

- Effectuez les modifications.

```
SQL> UPDATE   emp
  2  SET      deptno = 10
  3  WHERE    empno = 7782;
1 row updated.
```

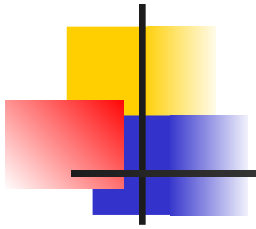
- Validez les modifications.

```
SQL> COMMIT;
Commit complete.
```

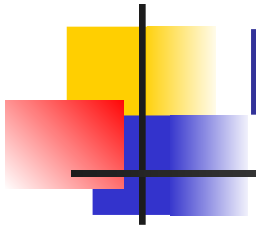
Etat des Données Après ROLLBACK

- L'ordre ROLLBACK rejette toutes les modifications de données en instance.
  - Les modifications sont annulées.
  - L'état précédent des données est restauré.

```
SQL> DELETE FROM    employee;
14 rows deleted.
SQL> ROLLBACK;
Rollback complete.
```



Contrôle des Accès Utilisateur

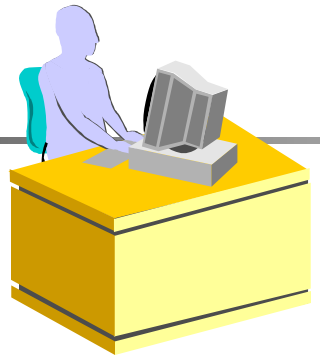


BD Multi Utilisateurs

- Une base de données est destinée à être exploitée par plusieurs utilisateurs.
- Le SGBD offre des outils pour gérer les utilisateurs, assurer la sécurité des données et garantir la confidentialité des informations.

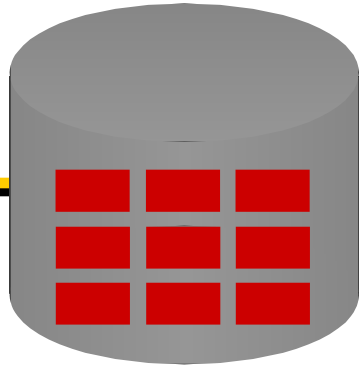
Contrôle des Accès Utilisateur

Administrateur de base de données



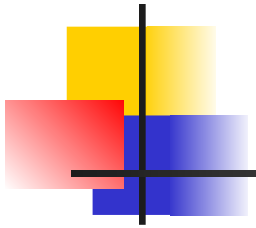
Privilèges

nom utilisateur et mot de passe



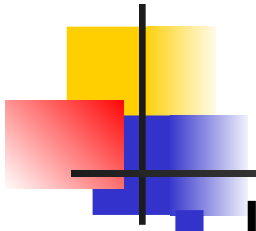
Utilisateurs

Partage des données entre utilisateurs



- Une base de données doit être accessible par tous les utilisateurs qui ont besoin des données stockées dans les différentes tables.
- Chaque utilisateur a accès à un sous ensemble des données (permissions ou de droits) qu'il partage avec d'autres utilisateurs.
- Le LCD (Langage de Contrôle des Données) regroupe les outils de mise en œuvre de la sécurité et des droits d'accès.

Sessions

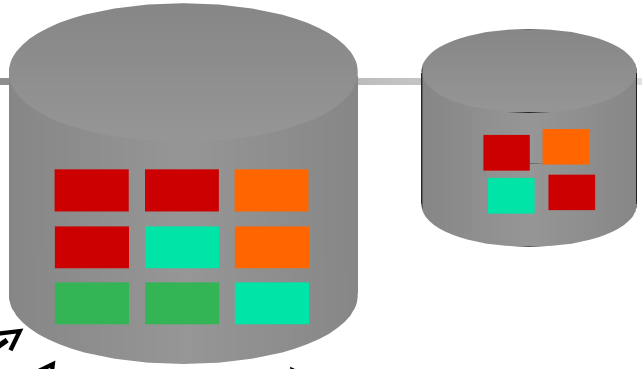


- Une session = une connexion à un SGBD par un utilisateur.
- L'utilisateur est identifié par:
  - login
  - mot de passe
- L'utilisateur possède
  - Un schéma = ens. d'objets : Bases de données, tables, vues
  - Privilèges ou Droits : pour modifier et visualiser des objets des bases de données appartenant à d'autres schémas

Contrôle des accès utilisateurs

Administrateur de BD

BDs



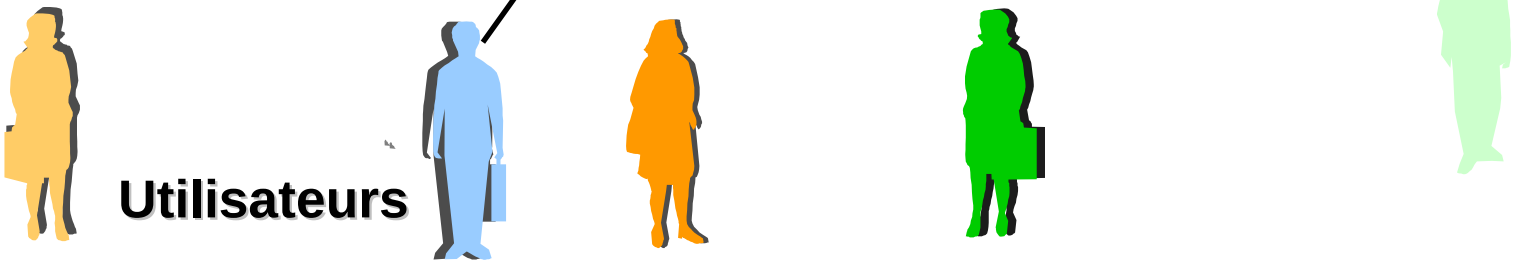
Sessions

Login,
Mot de passe
privilèges

Login,
Mot de passe
privilèges

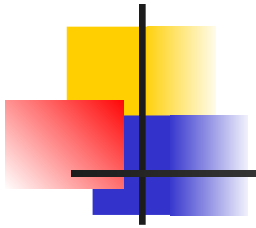
Login,
Mot de passe
privilèges

Login,
Mot de passe
privilèges



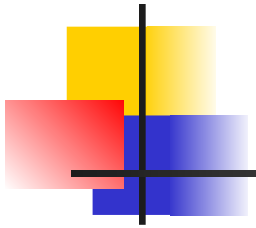
Utilisateurs

Propriétaire et schéma



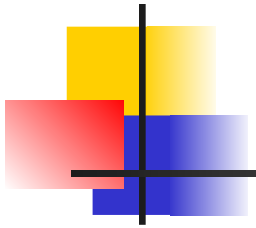
- Tout objet d'une BD appartient à un seul utilisateur (owner).
- Le propriétaire de l'objet correspond à l'utilisateur qui l'a créé.
- Le propriétaire a le droit d'accorder et de refuser l'accès à ses objets aux autres utilisateurs de la base de données.
- L'ensemble des objets créés par un utilisateur compose son schéma
  - Schéma = ens. de tables et de vues.

Privilèges



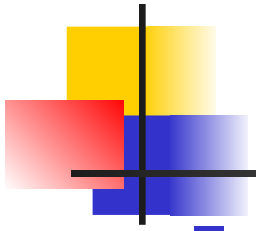
- Sécurité de la base de données
  - Sécurité du système
  - Sécurité des données
- Privilèges système : autorisent l'accès à la base de données
- Privilèges objet : autorisent la manipulation du contenu des objets de la base de données
- Schéma : collection d'objets, tels que les tables, les vues et les séquences

Droits ou privilèges



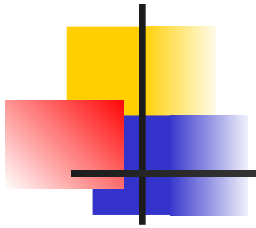
- Les droits accordés à chaque utilisateur sur les objets sont définis en fonction :
  - des besoins de l'utilisateur
  - des compétences de ce dernier
- Administrateur de la BD
  - possède tous les privilèges systèmes
  - Attribut les privilèges systèmes aux utilisateurs

Droits ou Privilèges



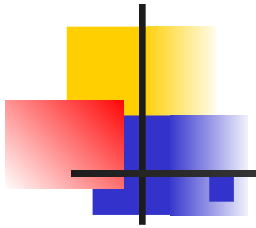
- Sécurité des bases de données
  - Sécurités du système (des schémas)
  - Sécurités des données
- Privilèges systèmes
  - Ajouter/modifier/supprimer des nouveaux objets
- Privilèges objets
  - Accéder/modifier/supprimer les données contenues dans les objets

Privilèges Système



- Plus de 80 privilèges sont disponibles.
- L'administrateur de base de données est doté de privilèges système de haut niveau.
  - Création de nouveaux utilisateurs
  - Suppression d'utilisateurs
  - Suppression de tables
  - Sauvegarde des tables

Privilèges systèmes



- Plusieurs types de privilèges systèmes sont possibles
  - CREATE
  - ALTER ...
- Les privilèges systèmes permettent d'ajouter ou de modifier des objets:
  - des bases de données,
  - des tables,
  - des vues,
  - des procédures
- L'administrateur DBA à tous les privilèges systèmes

Attribution de privilèges systèmes

```
GRANT  privilege_sys [, privilege_sys...]
TO     user [, user...];;
```

- Le DBA peut attribuer le droit de créer une base de données à l'utilisateur Amal.

```
SQL> GRANT  create database
  2  TO     amal;
Grant succeeded.
```

- Le DBA peut attribuer le droit de créer une table ou une vue à l'utilisateur Bachir.

```
SQL> GRANT  create table, create view
  2  TO     bachir;
Grant succeeded.
```

ALL et PUBLIC

- Attribution de tous les privilèges systèmes à l'utilisateur Jamil.

```
SQL> GRANT  ALL  TO  jamil;
```

- Attribution de tous les privilèges systèmes à tous les utilisateurs

```
SQL> GRANT  ALL  TO  PUBLIC;
```

- Administrateur de la BD
  - possède tous les privilèges systèmes
  - Attribut les privilèges systèmes aux utilisateurs

Privilèges objets

- Pour accéder/modifier/supprimer les données contenues dans les objets
- Les privilèges varient d'un objet à l'autre.
- Le propriétaire d'un objet à tous les droits/privilèges sur lui.
- Le propriétaire d'un objet peut accorder des privilèges objets à d'autres utilisateur sur cet objet.

```
GRANT        objet_priv [(colonnes)]
ON           nom_objet
TO           {utilisateur|PUBLIC}
[WITH GRANT OPTION];
```

Attribution des privilèges objets aux utilisateurs

- Attribution du privilège de rechercher des données sur la table EMP.

```
SQL> GRANT    select
  2  ON       emp
  3  TO       sue, rich;
Grant succeeded.
```

- Attribution du privilège de mettre à jour des colonnes particulières .

```
SQL> GRANT    update (dname, loc)
  2  ON       dept
  3  TO       scott, manager;
Grant succeeded.
```

Clause WITH GRANT OPTION

- Autorise l'utilisateur à attribuer ses privilèges à d'autres utilisateurs.
- Attribution du privilège objet SELECT et INSERT sur la table dept à l'utilisateur scott avec autorisation d'attribuer ces droits à d'autres utilisateurs.

```
SQL> GRANT    select, insert
  2  ON       dept
  3  TO       scott
  4  WITH GRANT OPTION;
Grant succeeded.
```

Retirer des privilèges

- Tout privilège système ou d'objet attribué à un utilisateur, à un groupe d'utilisateurs ou à tous les utilisateurs peut être retiré.
- Tout privilège attribué ou non par la clause GRANT OPTION peut être retiré.

```
REVOKE {privilege [, privilege...]|ALL}
ON     object
FROM   {user[, user...]|role|PUBLIC};
```

Retirer des privilèges systèmes

- Retirer le privilège système de création d'une base à l'utilisateur Amal.

```
SQL> REVOKE   create database
  2  FROM     amal;
Revoke succeeded.
```

- Retirer des privilèges systèmes à l'utilisateur Bachir.

```
SQL> REVOKE   create table, create view
  2  FROM     bachir;
Revoke succeeded.
```

Retirer des privilèges objets

- Retirer le droit de chercher et insérer des données dans la table DEPT à scott

```
SQL> REVOKE   select, insert
  2  ON       dept
  3  FROM     scott;
Revoke succeeded.
```

```
SQL> REVOKE   ALL
  2  ON       dept
  3  FROM     PUBLIC;
Revoke succeeded.
```

Création d'un Utilisateur

- L'ordre CREATE USER permet à l'administrateur de base de données de créer des utilisateurs

```
CREATE USER     user
IDENTIFIED BY  password;
```

```
SQL> CREATE  USER  scott
  2  IDENTIFIED BY tiger;
User created.
```

Privilèges Système de l'Utilisateur

- Une fois l'utilisateur créé, l'administrateur de base de données peut lui accorder des privilèges système particuliers.

```
GRANT privilege [, privilege...]
TO user [, user...];
```

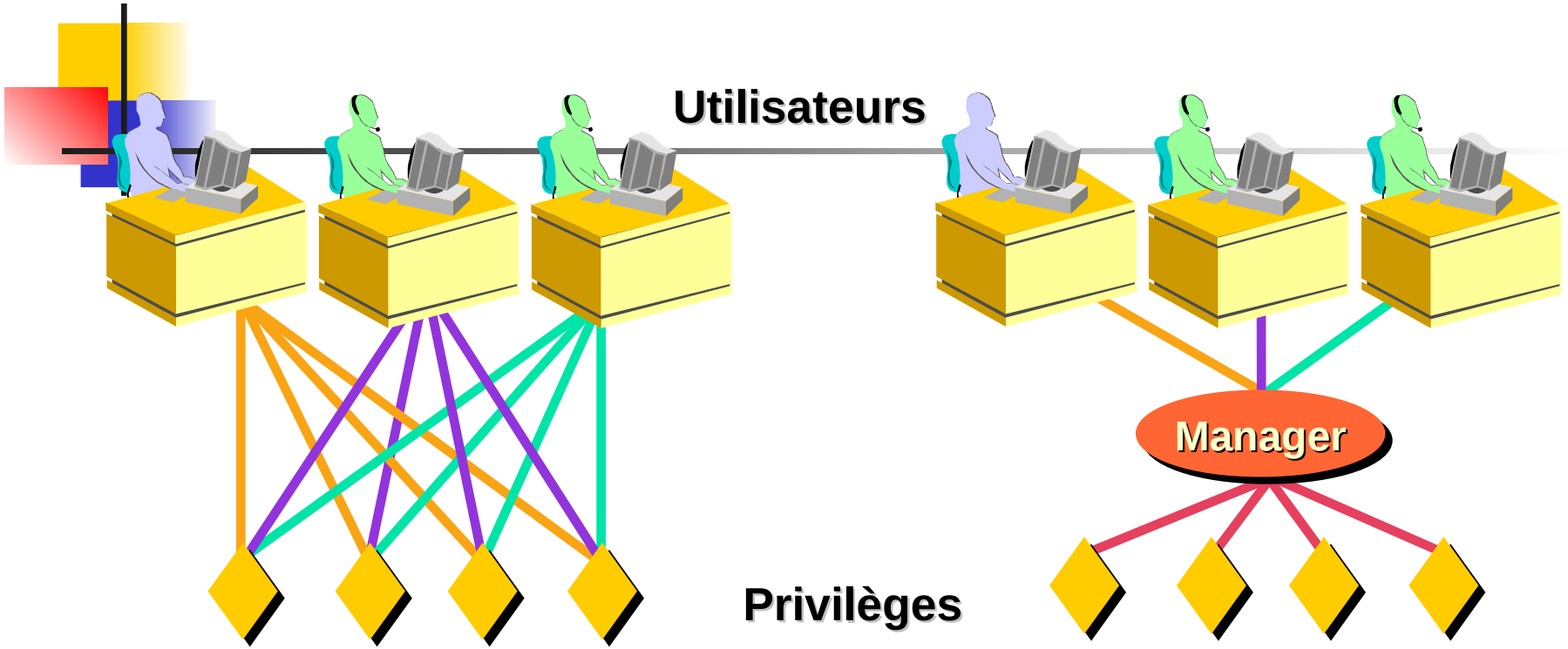
- Les privilèges système suivants peuvent être accordés à un développeur d'applications :
  - CREATE SESSION
  - CREATE TABLE
  - CREATE SEQUENCE
  - CREATE VIEW
  - CREATE PROCEDURE

Octroi de Privilèges Système

- L'administrateur de base de données peut accorder à un utilisateur des privilèges système particuliers.

```
SQL> GRANT  create table, create sequence, create view
  3  TO     scott;
Grant succeeded.
```

Qu'est-ce qu'un Rôle ?



Attribution de privilèges sans rôle

Attribution de privilèges avec un rôle

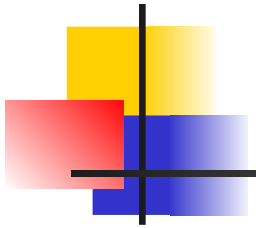
Création d'un Rôle

```
SQL> CREATE ROLE manager;
Role created.
```

```
SQL> GRANT create table, create view
2           to manager;
Grant succeeded.
```

```
SQL> GRANT manager to BLAKE, CLARK;
Grant succeeded.
```

Modification du Mot de Passe



- Lorsqu'un compte utilisateur est créé, un mot de passe est initialisé.
- Les utilisateurs peuvent modifier leur mot de passe à l'aide de l'ordre ALTER USER.

```
SQL> ALTER USER scott
  2         IDENTIFIED BY lion;
User altered.
```

Privilèges Objet

- Les privilèges objet varient d'un type d’objet à l'autre.
- Un propriétaire bénéficie de tous les privilèges sur ses objets.
- Un propriétaire peut accorder des privilèges particuliers sur les objets qui lui appartiennent.

```
GRANT        {object_priv [(columns)]|ALL}
ON           object
TO           {user|role|PUBLIC}
[WITH GRANT OPTION];
```

Octroi de Privilèges Objet

- Accordez des privilèges de lecture sur la table EMP.

```
SQL> GRANT    select
  2  ON       emp
  3  TO       sue, rich;
Grant succeeded.
```

- Accordez des privilèges aux utilisateurs et aux rôles pour mettre à jour des colonnes spécifiques.

```
SQL> GRANT    update (dname, loc)
  2  ON       dept
  3  TO       scott, manager;
Grant succeeded.
```

Utilisation des Mots-Clés WITH GRANT OPTION et PUBLIC

- Autorisez un utilisateur à transmettre des privilèges.

```
SQL> GRANT    select, insert
  2  ON       dept
  3  TO       scott
  4  WITH GRANT OPTION;
Grant succeeded.
```

- Autorisez tous les utilisateurs du système à interroger les données de la table DEPT d'Alice.

```
SQL> GRANT    select
  2  ON       alice.dept
  3  TO       PUBLIC;
Grant succeeded.
```

Comment Retirer les Privilèges Objet

- Retirez les privilèges accordés à d'autres utilisateurs à l'aide de l'ordre REVOKE.
- Les privilèges accordés avec le mot-clé WITH GRANT OPTION seront aussi

```
REVOKE {privilege [, privilege...]|ALL}
ON     object
FROM   {user[, user...]|role|PUBLIC}
[CASCADE CONSTRAINTS];
```

Retrait des Privilèges Objet

- En tant qu'utilisateur Alice, retirez les privilèges SELECT et INSERT accordés à l'utilisateur Scott dans la table DEPT.

```
SQL> REVOKE   select, insert
  2  ON       dept
  3  FROM     scott;
Revoke succeeded.
```